BUG SUMMARIZATION, SEVERITY CLASSIFICATION AND BUG ASSIGNMENT FOR AUTOMATED BUG RESOLUTION

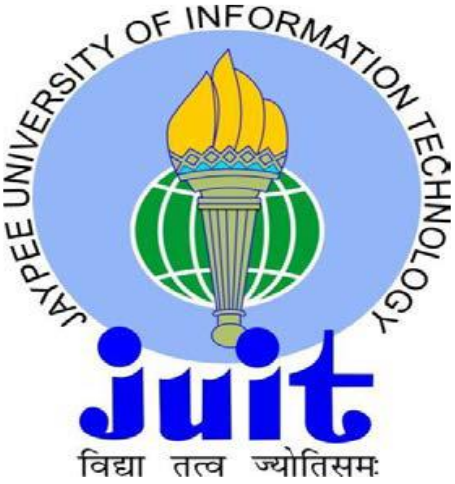
Synopsis submitted in fulfillment of the requirements for the Degree of

DOCTOR OF PHILOSOPHY

By

ASHIMA

Enrolment No. 166202



Under the Supervision of

Dr. Yugal Kumar and Dr. Rajni Mohana

Department of Computer Science Engineering and Information Technology

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

Waknaghat, Solan-173234, Himachal Pradesh, INDIA

August, 2019

1. Introduction

Software maintenance phase is an important aspect of Software Development Life Cycle (SDLC). The purpose of the software maintenance phase is to fix the software bugs and improves the quality of software. A software bug can be described as fault, mistake, and error. It is human error that can feed unintentionally during the execution of the software especially either in source code or design. If, these bugs are not detected in earlier phase of SDLC, then it can lead to failure of software and sometime unexpected behavior of software [1]. In turn, development cost of software's is increased and further also affected the delivery of software's. It is noticed that more than 90% of software development cost is spent on software evolution and maintenance activities [2]. In past few years, software maintenance can get wide attention from the research community. The main reasons are security risk, user inconvenience, inappropriate functionality, and large number of bugs. To handle aforementioned issues, bug tracking or reporting system are designed such as Mozilla, Eclipse, OpenFoam, JIRA, Trac, BugZilla, Redmine, OTRS, Mantis, FogBugz, BugNet etc. These systems detect the bugs in earlier phase of SDLC, prioritize the bugs and also ensure the timely delivery of software. The core component of bug tracking system (BTS) is bug repository. It contains various artifacts related to software's such as change history of status, source code and bug reports etc. The artifacts are used in software maintenance phase to resolve the bugs and also the key points for developers. To remove the bugs from software is known as bug resolution process (BRP). This process is handled through triager or developer. In BRP, bug reports have significant impact. It contains diverse information like freeform text, attachment and predefined fields. Freeform text contains the description, summary and comments. Attachments consist of non-textual information such as test cases, patches, etc. The predefined fields contain the various metadata, like status, product, importance, component, and assignee. Metadata is used to determine relevant features from bug report [3]. Further, the status in the predefined fields indicates the software bug life cycle (SBLC) [4-5]. The different statuses of SBLC are demonstrated in Figure 1.

- UNSOLVED: Initially, status of all bug reports is UNSOLVED.
- NEW: When, triager searches the appropriate developer for assigning the bugs, then status of bugs is changed, called NEW.

- ASSIGNED: Once, bugs are assigned to a relevant developer, then the status is ASSIGNED.
- RESOLVED: When, bugs are resolved from the source code, its status is RESOLVED.
- REOPEN: If, the tester is not satisfied with the bug solution, then status of bug is REOPEN.
- VERIFIED: It is indicated that tester approves the bug solution.
- CLOSED: It indicates the removal of bug.

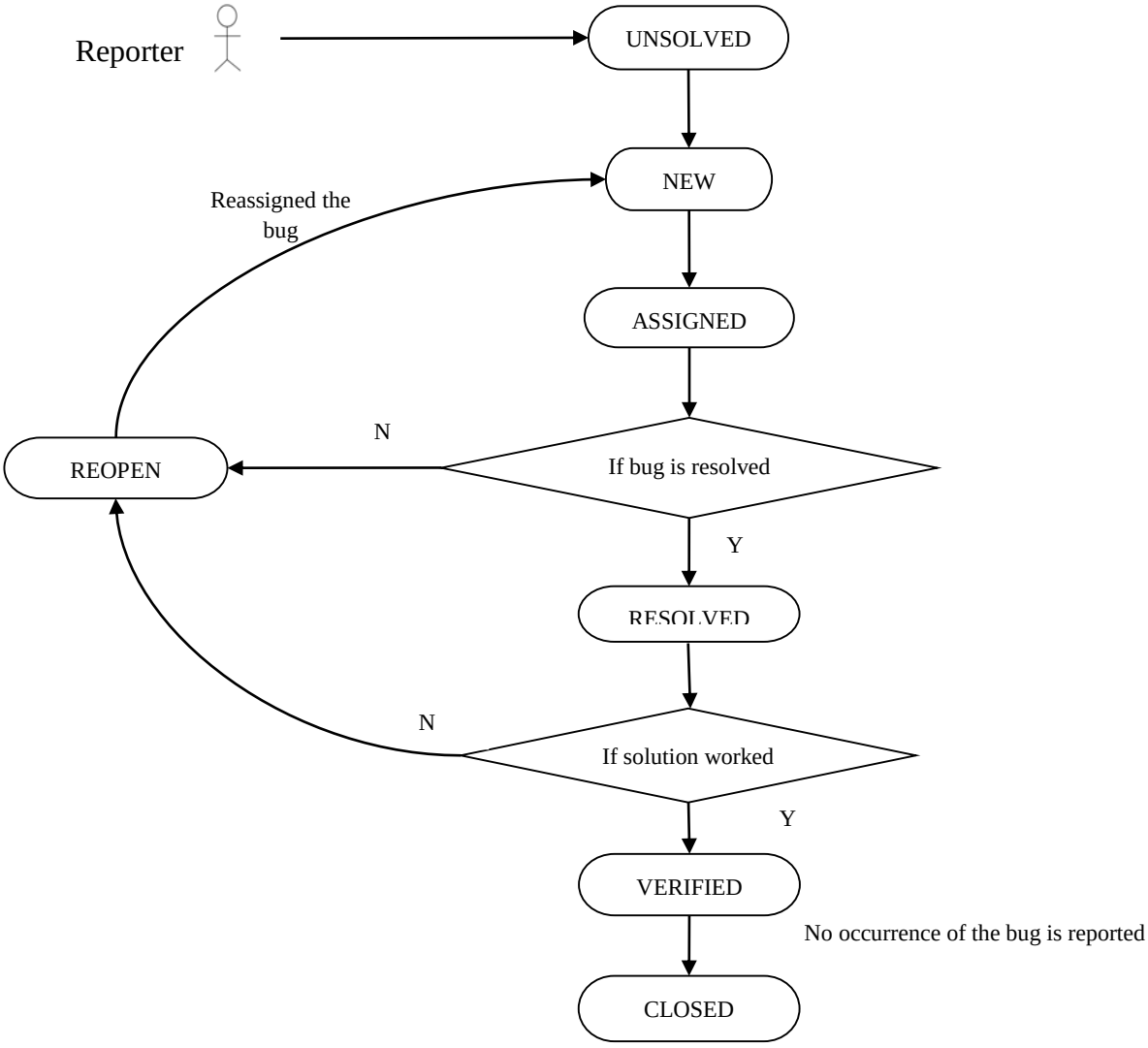


Figure 1: Software bug life cycle

BRP is iterative process and consists of three phases. These phases are bug report understanding, bug triage and bug fixing.

- Bug Report Understanding Phase: In this phase, the status is changed from NEW to ASSIGNED. Triagers explore the contents of given bug reports to summarize the content of bug report, filter the duplicate bugs and predict the features (e.g., priority, severity and reopened/blocking). Large number of bug reports is generated every day and to summarize these bug reports manually is one of challenging task [6]. This issue attracts various researcher to work in this direction i.e. developed automated tool for bug report understanding [7, 15-23].
- Bug Triaging Phase: This phase responsible to identify the appropriate developer to assign bugs. This can be done with the help of triagers. Without prior information on bug report, triagers cannot determine relevant developer to fix the bug. Hence, bug assignment is also an important task to improve the bug fixing rate. To address the bug assignment issue, several bug triage approaches have been developed for fixing the bugs [24-29].
- Bug Fixing Phase: In this phase, bugs are fixed with the help of developers. For a given bug, developer analyzes the source code files, identifies the source of bugs and also updates the patch code as a bug-fixing process. The manual bug localization and patch generation can make the bug fixing process difficult. Several researchers have also developed automatic tools to address bug fixing issue [30-32].

Bug resolution problem attracts wide attention from research community from past few years [7]. Research community provides both theoretical and experimental solution for bug resolution problem such as bug report summarization, identification of appropriate developer for bug fixing, bug severity classification etc. In literature, large numbers of supervised and unsupervised methods are reported for bug report summarization [7, 11-15]. Rastkar et al., [7] developed email classifier, bug report classifier, email meeting classifier with logistic regression for summarization task. Mani et al., [12] applied various page ranking method like centroid, grasshoper, diverse rank, and maximinum marignal relevance to select the sentences for summarization task. Lotufo et al., [13] also used page rank and markov chain methods to rank the sentences for generating the summaries. To handle the bug severity, large numbers of techniques have been presented for severity classification [16-23]. Zhou et al., [16] used text preprocessing methods with multinomial naive bayes classifier to classify the bugs into two classes i.e. bug and non-bug. Jin et al., [17] used preprocessing methods with naive bayes classifier to classify the data into severe and non-severe classes. Mishra et al., [21] applied bagging, vote technique, info gain, TF-IDF for feature selection

with naive bayes (NB), support vector machine (SVM) and K-nearest neighbors (K-NN) classifier to classify the bugs into Blocker, Critical, Major, Minor and Trivial classes. Bug Assignment is the process of assigning the relevant developer for bug fixing. The aim is to reduce the probability of reassigning of the bugs. It also reduces the triages time and effort to assign relevant bug fixer or developer. But, to determine relevant developer and rank is one of challenging task. Therefore, many techniques are proposed to determine the relevant developers [24-29]. Zhang et al., [24] used LDA for topic extraction from the previous bug reports and analysis the developer's relationship on the same topic using social network. The extracted features are used to rank the developers. Moreover, Yadav et al., [29] used Jaccard, cosine –similarity to measure the similarity between fixing time, priority and versatility, based on extracted features the developers are ranked. Hence, through extensive literature review, several shortcomings are identified. These shortcomings are given as

- Scope of improving the methods to handle bug report summarization, bug fixing etc.
- Scope to improvement the accuracy rate of bug severity classification.
- Identification of appropriate developer for bug resolution.
- Handle manual patch generation and bug localization as number of bugs are increased.

2. Objectives of the Thesis

The aim of this thesis is to resolve bug resolution issues through bug report summarization, feature extraction, accuracy rate of severity prediction, and identification of relevant developer. Hence, the objectives of this work are given as

- To implement domain specific Bug Report Summarization technique
- To improve the accuracy rate of bug severity classification model
- To develop a model for bug assignment.

3. Outline of Thesis

3.1. PSO-ACO Based Summarization Model

A PSO-ACO based subset selection technique is applied to determine the optimal subset of summaries. Steps of the proposed technique are mentioned in Algorithm 1. Figure 2 demonstrates the proposed model for bug report summarization. The working of proposed model is divided into four phases. In phase 3, PSO-ACO based technique is applied to determine optimal subset of summaries. The proposed PSO-ACO technique is implemented using Java platform. The

performance of the proposed technique is evaluated using ROUGE score [33]. Rastkar dataset [7,11] is used to validate the existence of proposed technique.

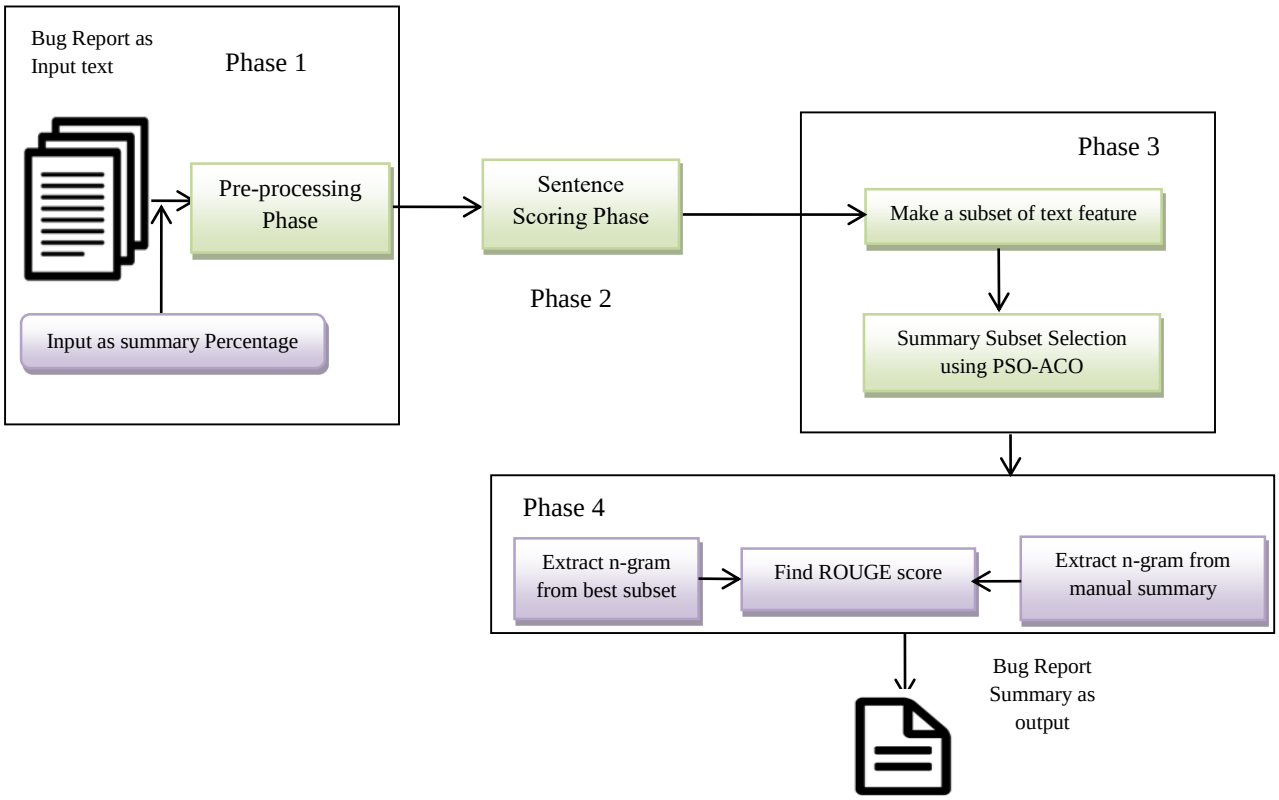


Figure 2: Proposed model for bug report summarization

The performance of PSO-ACO technique is compared with particle swarm optimization (PSO), grasshopper approach (EGA), bug report classifier (BRC), email classifier (EC) and email meeting classifier (EMC) [7, 11, 12]. Table 1 illustrates the simulation results of proposed technique and other algorithms.

Table 1: Comparison of PROPOSED PSO-ACO, PSO, BRC, EC, EMC and EGA techniques using average ROUGE score

Techniques	Average ROUGE score (%)		
	Summary Set 1	Summary Set 2	Summary Set 3
BRC	86.67	82.06	66.86
EC	84.69	55.39	51.71
EMC	85.54	61.29	39.03
EGA	74.17	68.13	63.54
PSO	91.32	86.47	82.97
PROPOSED PSO-ACO	92.60	87.92	84.51

Algorithm 1: Proposed PSO-ACO Based Subset Selection Algorithm

Input: Summary subset (m) with respective score

Output: Summary subset (n) with best optimized score, whereas n<m

Step 1: Initialize the particles and user define parameters viz population, iteration etc

Step 2: For, each subset or particle, d= 1,2…N
Compute fitness function and Particles initial position= subset score

Step 3: Subset best known position = initial position: $\mathrm{pbest_d} \leftarrow \mathrm{Pos_d^{(t)}}$

Step 4: Assume, initial velocity is zero

Step 5: While (condition is not met), do following:

Step 6: Update the Particle's velocity using equation 1

$$Vel_d^{(t+1)} = \alpha(t)Vel_d^t + \beta_{p1}rand_{p1}(t)\left(pbest_d - Pos_d(t)\right) + \beta_{p2}rand_{p2}(t)\left(gbest_d - Pos_d(t)\right) \quad (1)$$

Step 7: Update the Particle's position using equation 2

$$Pos_d^{(t+1)} = Pos_d^{(t)} + Vel_d^{(t+1)} \qquad (d = 1,2 \ldots . . N) \qquad (2)$$

Step 8: If $\mathrm{f(Pos_d^{(t)})} < \mathrm{f}\left(\mathrm{Pos_d^{(t+1)}}\right)$

$\mathrm{Pos_d^{(t+1)}} \leftarrow \mathrm{Pos_d(t)}$

Step 9: If $\mathrm{f\,(Pos_d^{(t+1)})} < \mathrm{f(pbest_d)}$

$\mathrm{pbest_{jd}} \leftarrow \mathrm{Pos_d^{(t+1)}}$

Step 10: If $\mathrm{f\,(pbest_d)} < \mathrm{f(gbest_d)}$

$\mathrm{gbest_d} \leftarrow \mathrm{pbest_d}$

End while

End for

Step 11: Sort the particles according the gbest and Calculate ROUGE score

Step 12: Initialize the ants and other user defined parameters of ACO algorithms

Step 13: Initialize the pheromones

Step 14: Update the pheromones

Step 15: If (value of subset is not changed)
Add the subset in subset list and calculate the transition probability
Select the subset with higher probability and calculate ROUGE score

Step 16: Else, update the pheromones of all subsets and go to step 13

Step 17: Obtain the final summary subset with best optimized score

*Vel_{id}^t is the old velocity, $Vel_d^{(t+1)}$ is the new velocity, $Pos_d^{(t)}$ represent as current position , $Pos_d^{(t+1)}$is updated particle position , $gbest_d$ is the global best position , $pbest_d$ is the personal best position, β_{p1}, β_{p2} are the learning factors as positive constant , $rand_{p1}(t), rand_{p2}(t)$ are random numbers between [0,1] and $\alpha(t)$ is the inertia weight .

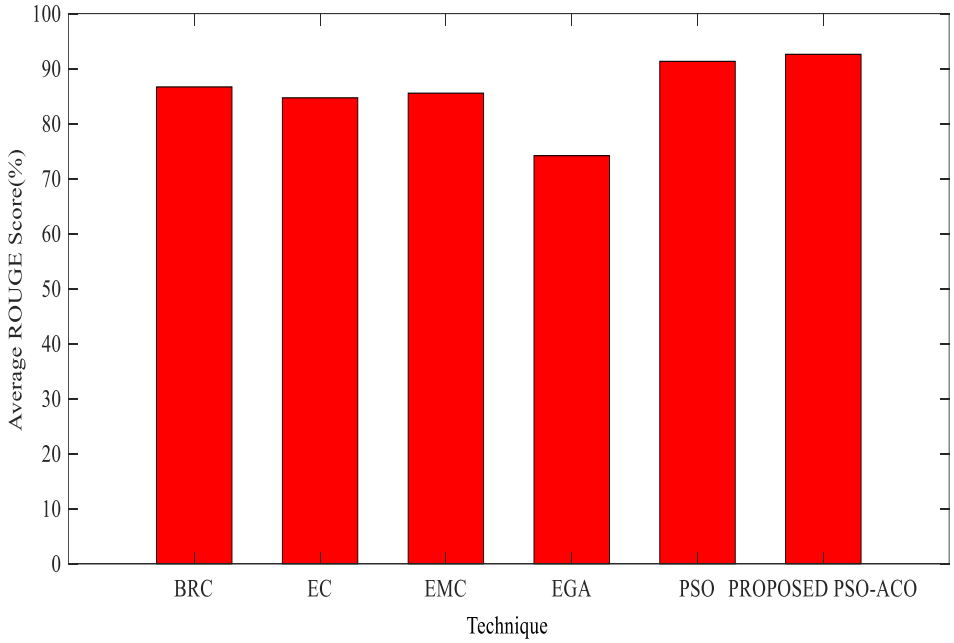


Figure 3: Simulation results of PROPOSED PSO-ACO, PSO, BRC, EC, ECM and EGA techniques on Summary Set 1

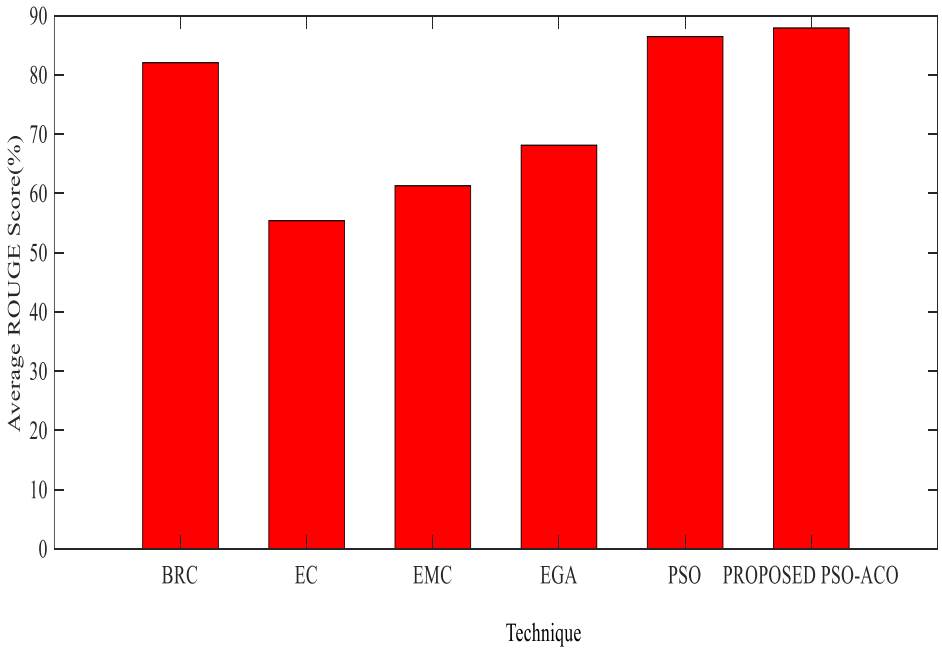


Figure 4: Simulation results of PROPOSED PSO-ACO, PSO, BRC, EC, ECM and EGA techniques on Summary Set 2

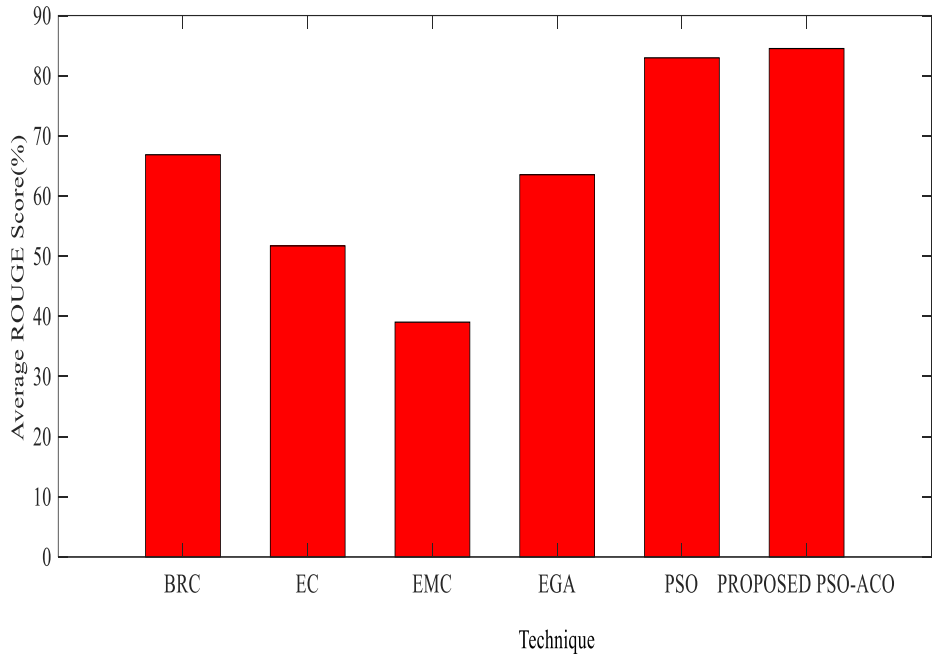


Figure 5: Simulation results of PROPOSED PSO-ACO, PSO, BRC, EC, ECM and EGA techniques on Summary Set 3

Figures 3-5 demonstrate the graphical representation of the simulation results of the proposed technique and other compared techniques using three summary sets. It is stated that proposed technique achieves better quality results than other techniques.

3.2. Random Forest with Boosting Method for Bug Severity Classification

The primary task of bug severity classification model is to classify the data in different severity classes. The classification model consists of machine learning approaches for predicting severity classes. To improve the prediction rate of classification model, an ensemble classifier is developed to classify the data into different classes. The proposed classifier is the combination of random forest and boosting method. Moreover, deep learning technique is applied to determine optimal set of features [34]. The proposed classification model for bug severity is illustrated in figure 6. Random forest with boosting method is applied for classification task. Steps of the proposed algorithm is mentioned in Algorithm 2. Five different datasets are considered to validate the existence of proposed model [28].

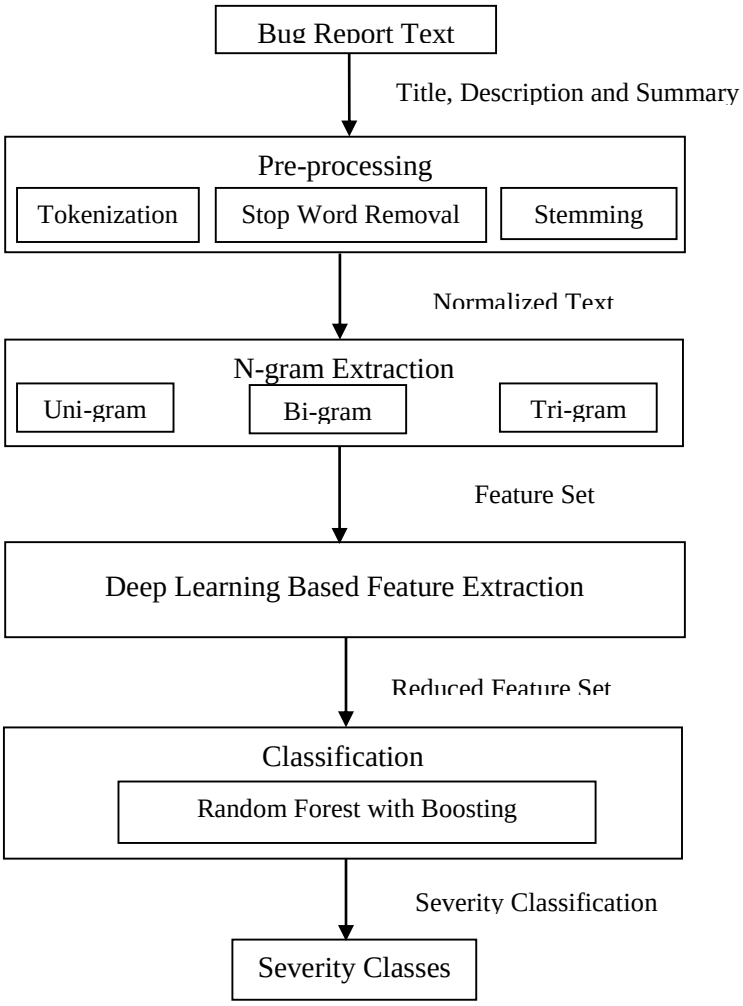


Figure 6: Proposed model for bug severity classification

Simulation results of the proposed model are evaluated using F-measure, Precision, Accuracy and Recall parameters [35]. Table 2 illustrates the simulation results of proposed model using five datasets. It is noticed that proposed model provides feasible and competent results for bug severity and achieves higher accuracy rate using all datasets. The performance of proposed model is also compared with the Zhou et al. model [16].

Figures 7-9 illustrates the simulation results of proposed model in graphical view using all five datasets. It is stated that proposed model achieves better quality results than Zhou et al., model.

Algorithm 2 : Random Forest with Boosting (RFB)

Input: Training samples with CN features and labels
Output: Trained RFB

Step 1: Randomly select "k" features from total "m" features, Where k << m

Step 2: Among the "k" features, calculate the node "d" using the best split point.

Step 3: Split the node into daughter nodes using the best split

Step 4: Repeat 1 to 3 steps until "l" number of nodes will not be reached

Step 5: Build forest by repeating steps 1 to 4 for "n" number times to create "n" number of decision trees

Step 6: Determine class label of training data

Step 7: Compute weighted error rate of decision tree

Step 8: Calculate the weight of decision tree's (w) using equation 3

$$w = {}^{1}/_{2} \log \frac{(M-1)(1-e)}{e} \quad (3)$$

Step 9: If (w >0)
Update weight of training samples

Step 10: Else reject the tree, repeat steps 7-10

* M is the number of classes, *e is the error rate of incorrect classified samples

Table 2: Experimental results of proposed model for severity classification using five datasets

Datasets	Classes	Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)
Mozilla	Blocker	82.9	80.23	81.54	85.45
	Critical	96.43	95.34	94.34	96.33
	Enhancement	95.33	80.22	95.45	97.43
	Major	98.45	97.45	92.34	96.3
	Normal	97.44	95.34	96.45	95.24
	Minor	94.35	96.34	97.44	96.34
	Trivial	96.35	95.34	95.33	97.33
Firefox	Blocker	86.16	83.35	84.73	83
	Critical	97.97	92.23	95.27	97.92
	Enhancement	98.3	92.23	95.98	97.55
	Major	97.98	97.61	96.64	97.19
	Normal	97.28	96.9	97.64	97.53
	Minor	97.23	97.04	97.24	98.33
	Trivial	98.43	95.68	96.52	98.85
Eclipse	Blocker	80.34	81.23	83.45	83.45
	Critical	97.33	89.23	96.35	98.33
	Enhancement	98.34	90.29	95.35	98.32
	Major	99.4	97.85	95.85	97.22
	Normal	97.35	97.29	98.4	97.24
Jboss	High	96.8	97.29	97.84	98.29

	Low	98.27	97	96.75	98.93
	Medium	99.08	95.45	96.72	99.22
	Unspecified	99.59	93.68	97.08	99.19
	Urgent	99.59	96.37	97.76	98.82
OpenFOAM	Blocker	83.45	82.34	83.45	83.4
	Crash	98.7	98.42	98.89	99.38
	Feature	99.28	97.81	98.33	98.56
	Major	99.61	98.51	97.11	99.54
	Minor	99.06	95.29	97.02	99.28
	Text	99.29	91.21	97.3	99.77
	Trivial	99.45	95.52	97.05	99.22
	Tweak	99.82	99.02	98.57	98.68

Table 3: Experimental results of proposed model and Zhou et al. model using five datasets

Datasets	**Zhou et al.**			**Proposed model**		
	Precision (%)	**Recall (%)**	**F-measure (%)**	**Precision (%)**	**Recall (%)**	**F-measure (%)**
Mozilla	82.60	82.40	81.70	98.48	98.52	98.12
Eclipse	81.80	82.10	81.60	99.38	99.48	99.12
JBoss	93.70	93.70	93.70	98.88	98.95	98.42
OpenFOAM	85.30	85.30	84.70	95.25	95.35	94.55
Firefox	80.30	80.50	79.50	92.75	92.95	91.95

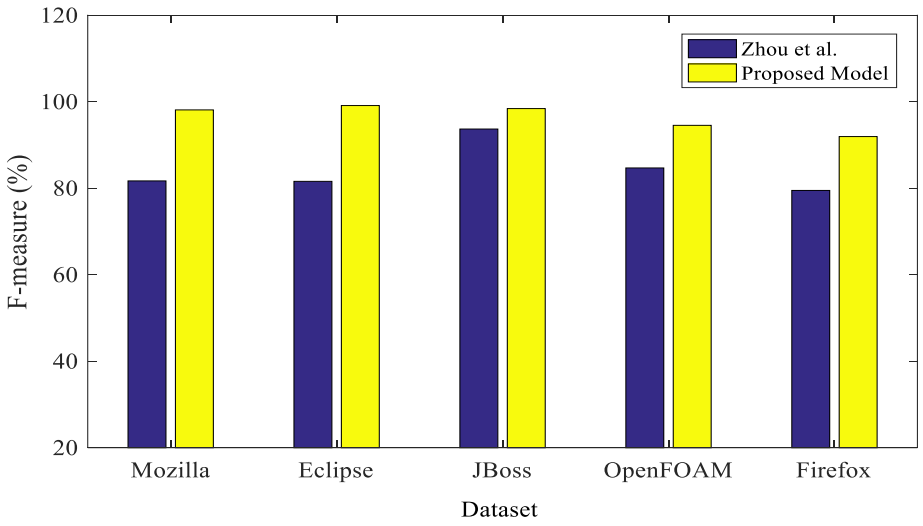


Figure 7: Comparison of proposed model and Zhou et al. model using F-measure parameter

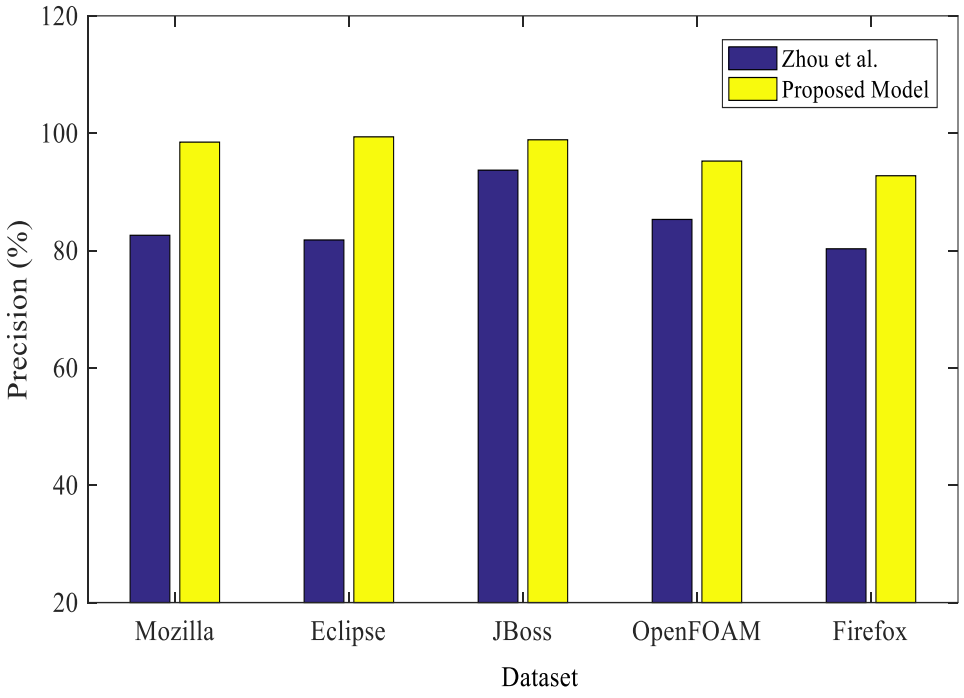


Figure 8: Comparison of proposed model and Zhou et al. model using precision parameter

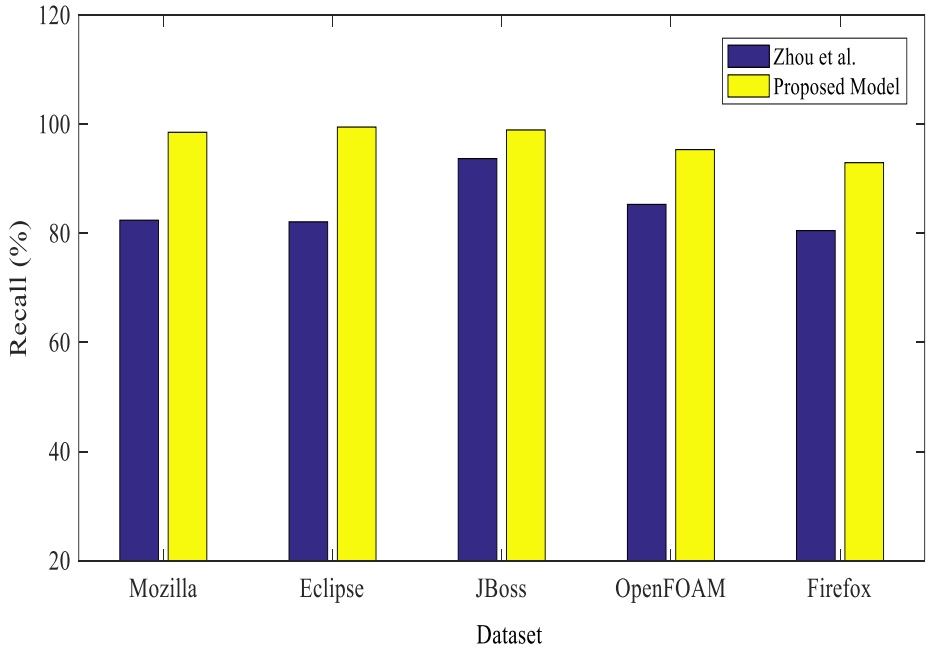


Figure 9: Comparison of proposed model and Zhou et al. model using recall parameter

3.3. ACO-NB and ACO-SVM Approaches Based Bug Assignment Model

To determine the set of relevant developers, a machine learning based bug assignment model is developed. The aim of this model is to assign the bugs to appropriate developers. The proposed model integrates the ACO-NB and ACO-SVM approaches. Initially, ACO approach is used to extract relevant features. Further, two machine learning techniques i.e. NB and SVM are applied to determine appropriate set of developers [36-38].

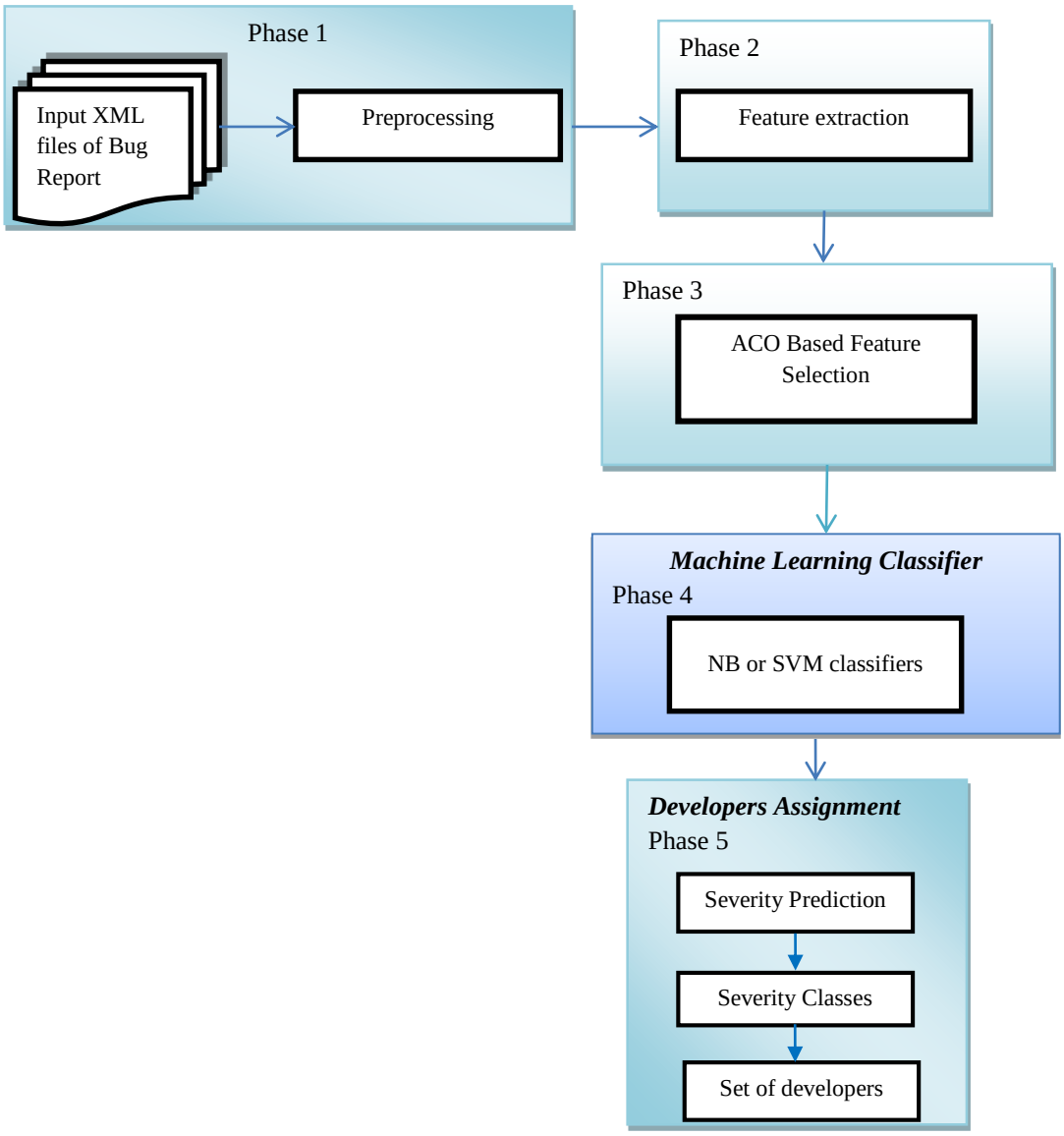


Figure 10: Proposed bug assignment model

These approaches predict the severity classes and on the basis of severity classes, set of relevant developers are identified to assign the bugs. The proposed bug assignment model is demonstrated

in Figure 10. Steps of the ACO based feature selection technique are mentioned in Algorithm 3. The performance of proposed model is evaluated using F-measure, Precision, Accuracy and Recall parameters [35]. Further, five different datasets are considered to evaluate the performance of proposed model [16]. Table 4 illustrates the simulation results of proposed model and other ML based bug assignment models using different datasets. It is seen that the performance of proposed ACO-NB and ACO-SVM based bug assignment model is better than existing machine learning models. Finally, it can be concluded that the proposed bug assignment model is capable to assign the set of relevant developers based on severity prediction.

Algorithm 3 : Proposed ACO Based Feature Selection

Input: Bug report dataset

Output: Reduced feature set

Step 1: Load the dataset and specify the numbers of initial clusters centers

Step 2: Initialize the population of ants, set the intensity of pheromone trial and maximum iteration.

Step 3: Randomly identified initial cluster centers for every ant

Step 4: Evaluate the value objective function using sum of squared distances and assign the items to the clusters with minimum objective function

Step 5: While (condition is not met)

Step 6: Calculate the transition probability using equation 4

$$p_A(T) = \frac{(\tau_A(T))^{\alpha}.\eta_A^{\beta}}{\sum_A(\tau_A(T))^{\alpha}.\eta_A^{\beta}} \quad (4)$$

Step 7: Update the pheromones using equation 5

$$\tau_A(T+1) = \rho\tau_A(T) + \Delta\tau_A(T) \quad (5)$$

Step 8: Determine the new positions of ants

Step 9: Recalculate the value of objective function

Step 10: End while

Step 11: Obtained the optimal clusters centers

Step 12: Determine the weights of each features (f_i) by using equation 6

$$f_i = \left(\sum_{i=1}^{d}\sum_{j=1}^{h}\sum_{k=1}^{k}\frac{X_{ih}}{C_{ik}}\right) \times \frac{1}{d} \quad (6)$$

Step 13: Select the features with maximum weight.

* η_β describe the background information of features to improve the results, $\tau_A(T)$ is pheromone amount for the Ath feature in time T, α and β are the control parameters that provides the pheromone and background information, and $p_A(T)$ is the Transition probability , ρ is defined as evaporation rate of pheromone trail and lies between [0, 1], $\Delta\tau_A(T)$ is the pheromone trail amount added to Ath feature between time Δ and ΔT , d is the total number of features, h is the number of data instance, k is the number of clusters, X_{ih} is the ith feature of hth data instance, C_{ik} is the ith feature of kth cluster

Table 4: Comparison of proposed bug assignment model, NB and SVM Classifiers for all five datasets

Approaches	Datasets	Performance Parameters			
		Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)
ACO-NB	Mozilla	78.75	77.51	78.61	80
	Eclipse	81.05	77.43	79.69	79.6
	JBoss	80.85	79.23	80.52	79.6
	OpenFOAM	77.59	76.85	77.71	78.25
	Firefox	77.04	77.22	77.62	79
NB	Mozilla	74.32	71.79	73.52	75.85
	Eclipse	72.85	71.22	72.52	73.4
	JBoss	74.05	70.42	72.69	74.8
	OpenFOAM	73.71	72.24	73.45	74.5
	Firefox	73.03	72.08	73.05	75.57
ACO-SVM	Mozilla	75.71	77.09	76	76.21
	Eclipse	73.33	72.08	72.53	74.72
	JBoss	73.65	72.56	72.62	75.24
	OpenFOAM	76.46	73.33	74.52	74.52
	Firefox	71.78	68.01	68.78	71.78
SVM	Mozilla	72.62	75.51	73.57	73.67
	Eclipse	73.56	68.73	70.89	71.67
	JBoss	73.95	69.12	70.91	71.67
	OpenFOAM	74.46	72.65	73.14	71.87
	Firefox	69.22	67.22	67.38	65.34

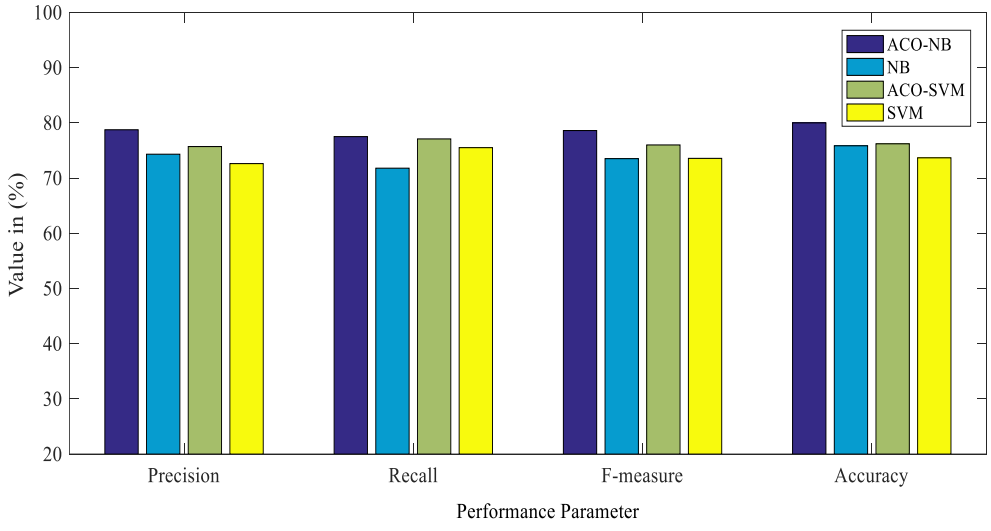


Figure 11: Performance comparison of proposed bug assignment model, NB and SVM classifiers using Mozilla dataset

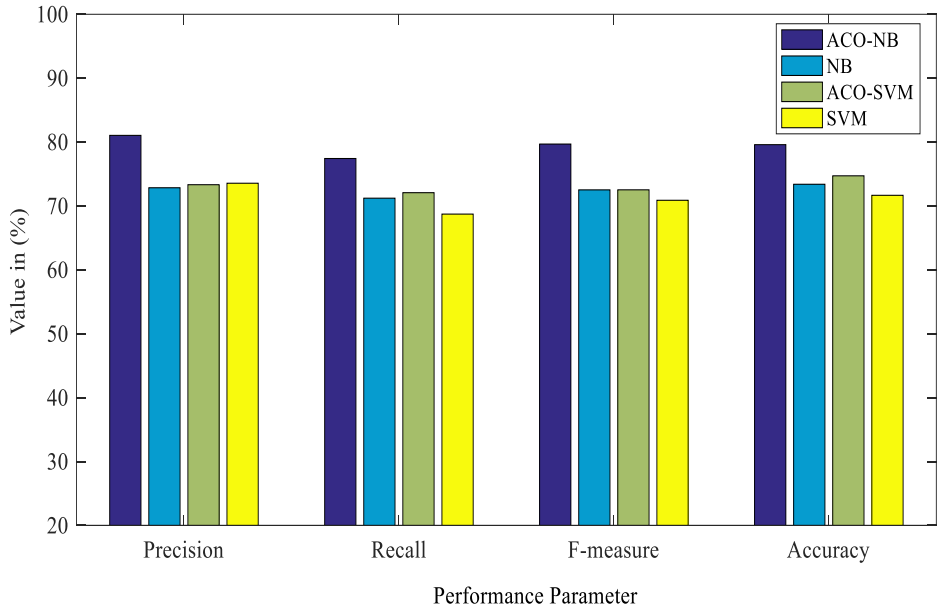


Figure 12: Performance comparison of proposed bug assignment model, NB and SVM classifiers using Eclipse dataset

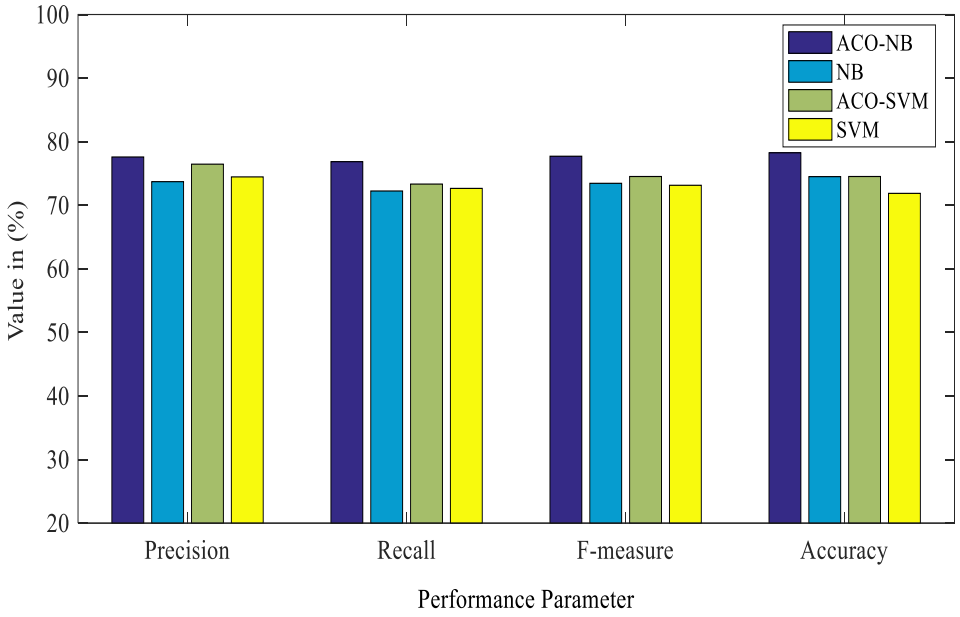


Figure 13: Performance comparison of proposed bug assignment model, NB and SVM classifiers using OpenFOAM dataset

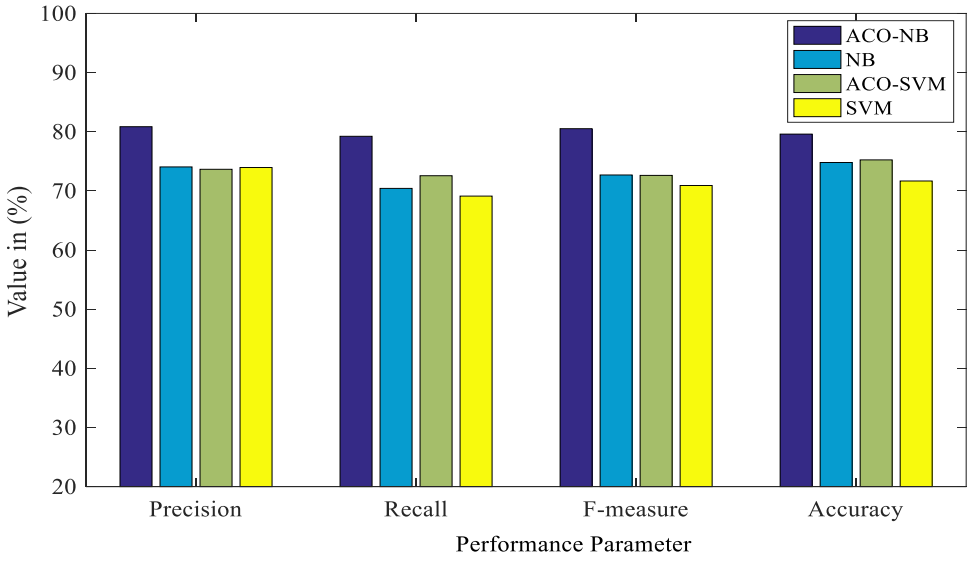


Figure 14: Performance comparison of proposed bug assignment model, NB and SVM classifiers using Jboss dataset

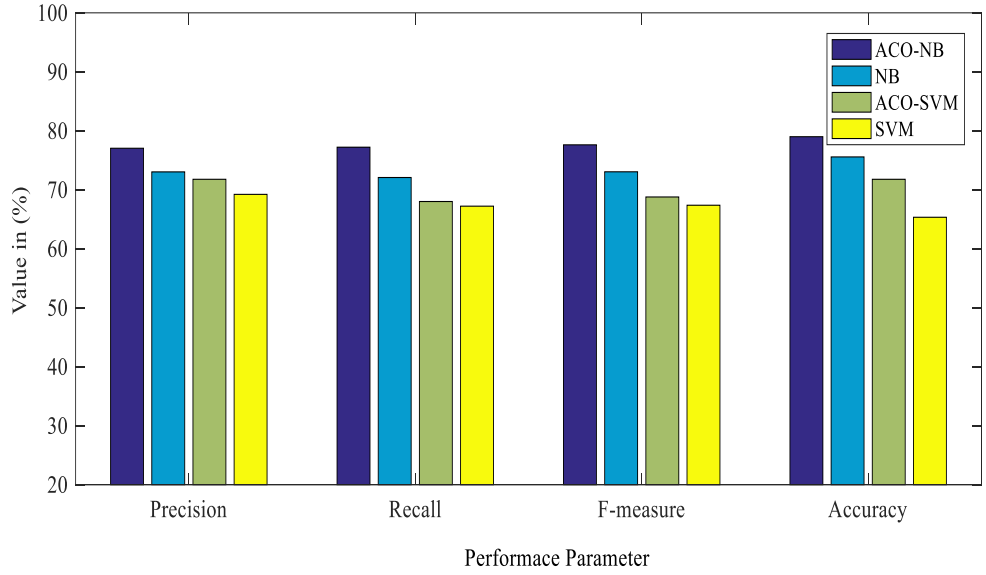


Figure 15: Performance comparison of proposed bug assignment model, NB and SVM classifiers using Firefox dataset

4. Conclusion

In this thesis, the bug resolution techniques are studied and categorized. The aim of work is to address features extraction, sparse representation of data, classification accuracy rate and identification of relevant developer issues of bug resolution process. The primary objectives of

this thesis are to implement domain specific bug report summarization technique, improve the accuracy rate of bug severity classification model and develop a model for bug assignment. To address the above mentioned issues, three different models are developed to address bug report summarization, bug severity classification and bug assignment issues. A PSO-ACO based bug report summarization technique is developed to handle bug report summarization issue. The aim of proposed technique is to determine the optimum subset of summary. Simulations results showed that PSO-ACO bases summarization technique provides better results than other existing algorithms. It is also observed that combination PSO and ACO enhances the simulation results of PSO technique in efficient manner.

To handle accuracy issue, random forest with boosting method is integrated into bug severity classification models. Moreover, deep learning technique is adopted for measuring the optimal set of features. Several benchmark datasets are considered to validate the performance of proposed model. Further, the simulation results of proposed model are compared with Zhou et al. model and it is stated that proposed model provides better results.

A bug assignment model is developed to identify the relevant set of developers for bug fixing. The set of developer are determined through severity classes. The proposed model integrates two techniques i.e. ACO-NB and ACO-SVM. The ACO technique is used to determine optimal features for severity prediction. While, NB and SVM techniques are applied for classification of severity data. Further, severity classes are used to determine relevant developers for bug fixing.

5. Future Scope

This thesis addresses the three issues of BRP process i.e. bug report summarization, severity classification and bug assignment. To determine the optimal set of features, ACO and PSO techniques is applied on bug report dataset. In future, some other meta-heuristic algorithms like ABC, Cuckoo search, Bat algorithm will be explored to determine optimal features. Further, NB, SVM and RFB classifiers are used to predict the severity classes and bug fixing. In future, some ensemble classifiers and meta-heuristic techniques will be considered for severity prediction and bug fixing. Furthermore, in future, bug duplicate detection, bug localization and patch generation issues will be taken into consideration and proposed new models to address these issues.

List of Publications

Journals

Accepted

1. Kukkar, A. and Mohana, R., 2018. Feature Weighting with for Swarm Intelligence Optimization as a Tool for Bug Report Summarization. Journal of Advanced Research in Dynamical and Control Systems, pp. 2122-2133**. (Published, Scopus)**
2. Kukkar, A. and Mohana, R. Bug Report Summarization by using Swarm Intelligence Approaches. Recent Patents on Computer Science. **(Accepted, Scopus)** DOI No:10.2174/2213275912666 181205154129.

Communicated

1. Kukkar, A. Mohana, R and Kumar, Y. Ant Colony Optimization based Model for Bug Triaging. Information and Software Technology **(SCI, Communicated)**
2. Kukkar, A. Mohana, R and Kumar, Y. An Ant Colony Optimization Based Feature Weighting Technique for Bug Severity Classification. International Journal of System Assurance Engineering and Management, Springer **(Scopus, Communicated)**

Conferences

1. Kukkar, A. and Mohana, R., 2018. A Supervised Bug Report Classification with Incorporate and Textual Field Knowledge. Procedia Computer Science, Elsevier 132, pp.352-361. **(Published, Scopus)**
2. Kukkar, A. and Mohana, R., 2019. An Optimization Technique for Unsupervised Automatic Extractive Bug Report Summarization. In International Conference on Innovative Computing and Communications (pp. 1-11). Springer, Singapore. **(Published, Scopus)**
3. Kukkar,A. Mohana, R and Kumar, Y., Does bug report summarization help in enhancing the accuracy of bug severity classification?. Procedia Computer Science, Elsevier **(Accepted, Scopus)**

Others

1. Kukkar, A., Mohana, R., Nayyar, A., Kim, J., Kang, B. G., & Chilamkurti, N. (2019). A Novel Deep-Learning-Based Bug Severity Classification Technique Using Convolutional Neural Networks and Random Forest with Boosting. Sensors, 19(13), 2964. **(SCI, Scopus, Published)**
2. Kukkar,A. Mohana, R and Kumar, Y., Duplicate Bug Report Detection using Deep Learning. **(under Submission)**

References

[1] Charette, R. N. (2005). Why software fails [software failure]. *IEEE spectrum*, *42*(9), 42-49.

[2] Erlikh, L. (2000). Leveraging legacy system dollars for e-business. *IT professional*, *2*(3), 17-23.

[3] Zhang, T., Jiang, H., Luo, X., & Chan, A. T. (2016). A literature review of research in bug resolution: Tasks, challenges and future directions. *The Computer Journal*, *59*(5), 741-773.

[4] Aljedaani, W., &Javed, Y. (2018). Bug Reports Evolution in Open Source Systems. In *5th International Symposium on Data Mining Applications* (pp. 63-73). Springer, Cham.

[5] Nazar, N., Hu, Y., & Jiang, H. (2016). Summarizing software artifacts: A literature review. *Journal of Computer Science and Technology*, *31*(5), 883-909.

[6] Eclipse. http://eclipse.org/.

[7] Rastkar, S., Murphy, G. C., & Murray, G. (2010, May). Summarizing software artifacts: a case study of bug reports. In *2010 ACM/IEEE 32nd International Conference on Software Engineering* (Vol. 1, pp. 505-514). IEEE.

[8] Antoniol, G., Ayari, K., Di Penta, M., Khomh, F., &Guéhéneuc, Y. G. (2008). Is it a bug or an enhancement? a text-based approach to classify change requests. In *CASCON* (Vol. 8, pp. 304-318).

[9] Angel, T. S., Kumar, G. S., Sehgal, V. M., & Nayak, G. (2018). Effective Bug Processing and Tracking System. *Journal of Computational and Theoretical Nanoscience*, *15*(8), 2604-2606.

[10] Thomas, S. W., Nagappan, M., Blostein, D., & Hassan, A. E. (2013). The impact of classifier configuration and classifier combination on bug localization. *IEEE Transactions on Software Engineering*, *39*(10), 1427-1443.

[11] Rastkar, S., Murphy, G. C., & Murray, G. (2014). Automatic summarization of bug reports. *IEEE Transactions on Software Engineering*, *40*(4), 366-380.

[12] Mani, S., Catherine, R., Sinha, V. S., & Dubey, A. (2012, November). Ausum: approach for unsupervised bug report summarization. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering* (p. 11). ACM.

[13] Lotufo, R., Malik, Z., & Czarnecki, K. (2015). Modelling the 'hurried'bug report reading process to summarize bug reports. *Empirical Software Engineering*, *20*(2), 516-548.

[14] Ferreira, I., Cirilo, E., Vieira, V., &Mourao, F. (2016, September). Bug report summarization: an evaluation of ranking techniques. In *2016 X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*(pp. 101-110). IEEE.

[15] Li, X., Jiang, H., Liu, D., Ren, Z., & Li, G. (2018, May). Unsupervised deep bug report summarization. In *Proceedings of the 26th Conference on Program Comprehension* (pp. 144-155). ACM.

[16] Zhou, Y., Tong, Y., Gu, R., & Gall, H. (2016). Combining text mining and data mining for bug report classification. *Journal of Software: Evolution and Process*, *28*(3), 150-176.

[17] Jin, K., Dashbalbar, A., Yang, G., Lee, J. W., & Lee, B. (2016). Bug severity prediction by classifying normal bugs with text and meta-field information. *Advanced Science and Technology Letters*, *129*, 19-24.

[18] Pandey, N., Sanyal, D. K., Hudait, A., & Sen, A. (2017). Automated classification of software issue reports using machine learning techniques: an empirical study. *Innovations in Systems and Software Engineering*, *13*(4), 279-297.

[19] Zhang, T., Yang, G., Lee, B., & Chan, A. T. (2015, April). Predicting severity of bug report by mining bug repository with concept profile. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing* (pp. 1553-1558). ACM.

[20] Jindal, R., Malhotra, R., & Jain, A. (2017). Prediction of defect severity by mining software project reports. *International Journal of System Assurance Engineering and Management*, *8*(2), 334-351.

[21] Singh, V. B., Misra, S., & Sharma, M. (2017). Bug severity assessment in cross project context and identifying training candidates. *Journal of Information & Knowledge Management*, *16*(01), 1750005.

[22] Sharmin, S., Aktar, F., Ali, A. A., Khan, M. A. H., &Shoyaib, M. (2017). Bfsp: A feature selection method for bug severity classification. In *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)* (pp. 750-754). IEEE.

[23] Kumari, M., Sharma, M., & Singh, V. B. (2018). Severity assessment of a reported bug by considering its uncertainty and irregular state. *International Journal of Open Source Software and Processes (IJOSSP)*, *9*(4), 20-46.

[24] Zhang, T., Yang, G., Lee, B., & Lua, E. K. (2014). A novel developer ranking algorithm for automatic bug triage using topic model and developer relations. In *2014 21st Asia-Pacific Software Engineering Conference* (Vol. 1, pp. 223-230). IEEE.

[25] Xuan, J., Jiang, H., Ren, Z., Yan, J., & Luo, Z. (2017). Automatic bug triage using semi-supervised text classification. *arXiv preprint arXiv:1704.04769*.

[26] Yang, G., Zhang, T., & Lee, B. (2014, July). Towards semi-automatic bug triage and severity prediction based on topic model and multi-feature of bug reports. In *2014 IEEE 38th Annual Computer Software and Applications Conference* (pp. 97-106). IEEE.

[27] Zhang, W., Wang, S., & Wang, Q. (2016). KSAP: An approach to bug report assignment using KNN search and heterogeneous proximity. *Information and Software Technology*, *70*, 68-84.

[28] Liu, J., Tian, Y., Yu, X., Yang, Z., Jia, X., Ma, C., & Xu, Z. (2016). A multi-source approach for bug triage. *International Journal of Software Engineering and Knowledge Engineering*, *26*(09n10), 1593-1604.

[29] Yadav, A., Singh, S. K., & Suri, J. S. (2019). Ranking of software developers based on expertise score for bug triaging. *Information and Software Technology*, *112*, 1-17.

[30] Rao, S., & Kak, A. (2011, May). Retrieval from software libraries for bug localization: a comparative study of generic and composite text models. In Proceedings of the 8th Working Conference on Mining Software Repositories (pp. 43-52). ACM.

[31] Wang, S., & Lo, D. (2014, June). Version history, similar report, and structure: Putting them together for improved bug localization. In Proceedings of the 22nd International Conference on Program Comprehension (pp. 53-63). ACM.

[32] Kim, D., Nam, J., Song, J., & Kim, S. (2013, May). Automatic patch generation learned from human-written patches. In Proceedings of the 2013 International Conference on Software Engineering (pp. 802-811). IEEE Press.

[33] Lin, C. Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74-81).

[34] Wei, Y., Zhao, Y., Lu, C., Wei, S., Liu, L., Zhu, Z., & Yan, S. (2016). Cross-modal retrieval with CNN visual features: A new baseline. *IEEE transactions on cybernetics*, *47*(2), 449-460.

[35] Huang, Y. J., Powers, R., &Montelione, G. T. (2005). Protein NMR recall, precision, and F-measure scores (RPF scores): structure quality assessment measures based on information retrieval statistics. *Journal of the American Chemical Society*, *127*(6), 1665-1674.

[36] Basiri, M. E., Ghasem-Aghaee, N., &Aghdam, M. H. (2008, March). Using ant colony optimization-based selected features for predicting post-synaptic activity in proteins. In *European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics* (pp. 12-23). Springer, Berlin, Heidelberg.

[37] Mitchell T (1996) Machine Learning. McCraw Hill

[38] Shi, Y., Tian, Y., Kou, G., Peng, Y., & Li, J. (2011). *Optimization based data mining: theory and applications*. Springer Science & Business Media.